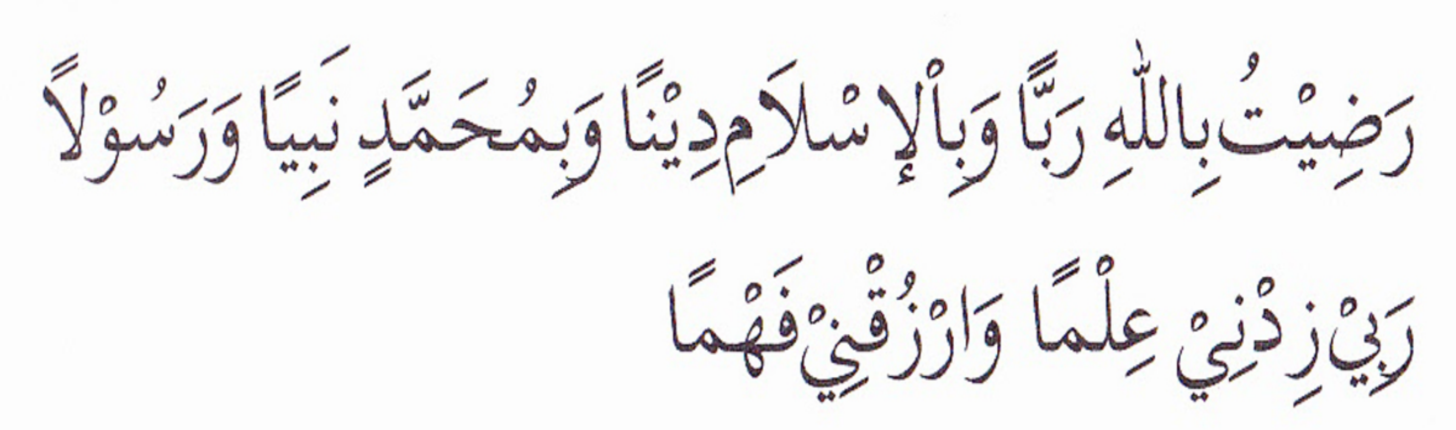
DOA SEBELUM BELAJAR



رَضِيْتُ بِاللهِ رَبًّا وَبِالْإِسْلَامِ دِيْنًا وَبِمُحَمَّدٍ نَبِيًّا وَرَسُوْلًا

رَبِيْ زِدْنِيْ عِلْمًا وَارْزُقْنِيْ فَهْمًا

“Kami ridho Allah SWT sebagai Tuhanku, Islam sebagai agamaku, dan Nabi Muhammad sebagai Nabi dan Rasul, Ya Allah, tambahkanlah kepadaku ilmu dan berikanlah aku kefahaman”

Universitas 'Aisyiyah
Yogyakarta

Pengantar Dasar Pemograman

Dasar Pemograman

Sadr Lufti Mufreni, S.Kom., M.Sc.

Sadr Lufti Mufreni, S.Kom., M.Sc.

1. SDN 2 Sukasari Tasikmalaya
2. SMP N 2 Tasikmalaya
3. SMA Muhammadiyah 1 Yogyakarta
4. S1 Ilmu Komputer Universitas Indonesia
5. S2 Teknologi Informasi Universitas Stuttgart

Learning Outcomes

Mahasiswa mampu memahami, dan menjelaskan sejarah bahasa, konsep dasar pemograman

Pesan hari ini

“Hai orang-orang yang beriman, jika datang kepadamu orang fasik membawa suatu berita, maka periksalah dengan teliti, agar kamu tidak menimpakan suatu musibah kepada suatu kaum tanpa mengetahui keadaannya yang menyebabkan kamu menyesal atas perbuatanmu itu”. [Al Hujurat : 6]

Outlines

- Perangkat Keras (Hardware)
- Sejarah Bahasa Pemograman
- Paradigma Pemograman
- Sekilas Tentang Bahasa Pemograman C

Perangkat Keras (Hardware)

- Hukum Moore : Perkembangan komputer akan meningkat dua kali lipat per tahun
- China's National University of Defense Technology's Tianhe-2 dapat melakukan 33.000.000.000.000.000 kalkulasi per detik
- Program komputer berisi kumpulan instruksi
- Pembuatnya disebut dengan programmer

Komputer

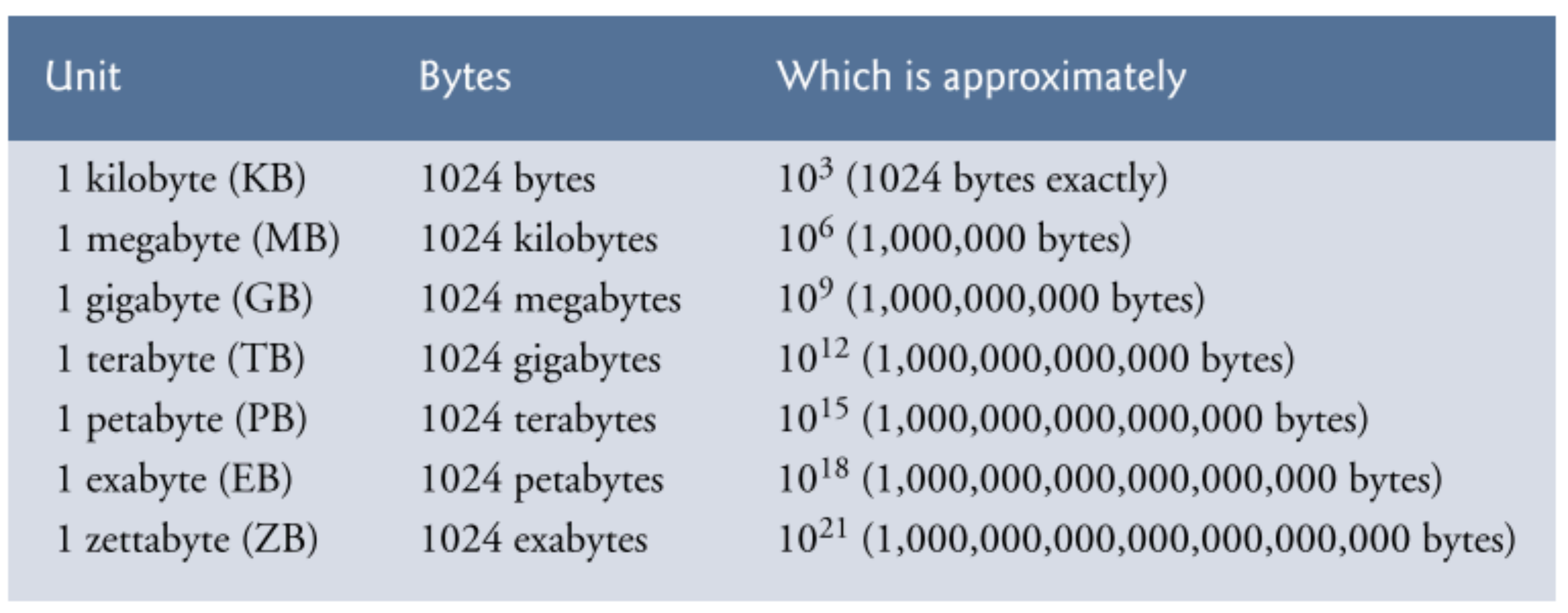
Komputer terdiri dari beberapa unit

1. Input
2. Output
3. Memori
4. Arithmetic and Logic Unit (ALU)
5. Central Processing Unit (CPU)
6. Media penyimpanan

Hirakri Data

Hirakri Data dari yang paling sederhana

1. Bits (0/1)
2. Karakter (ASCII -> Unicode)
3. Field
4. Record
5. File
6. Database
7. Big Data

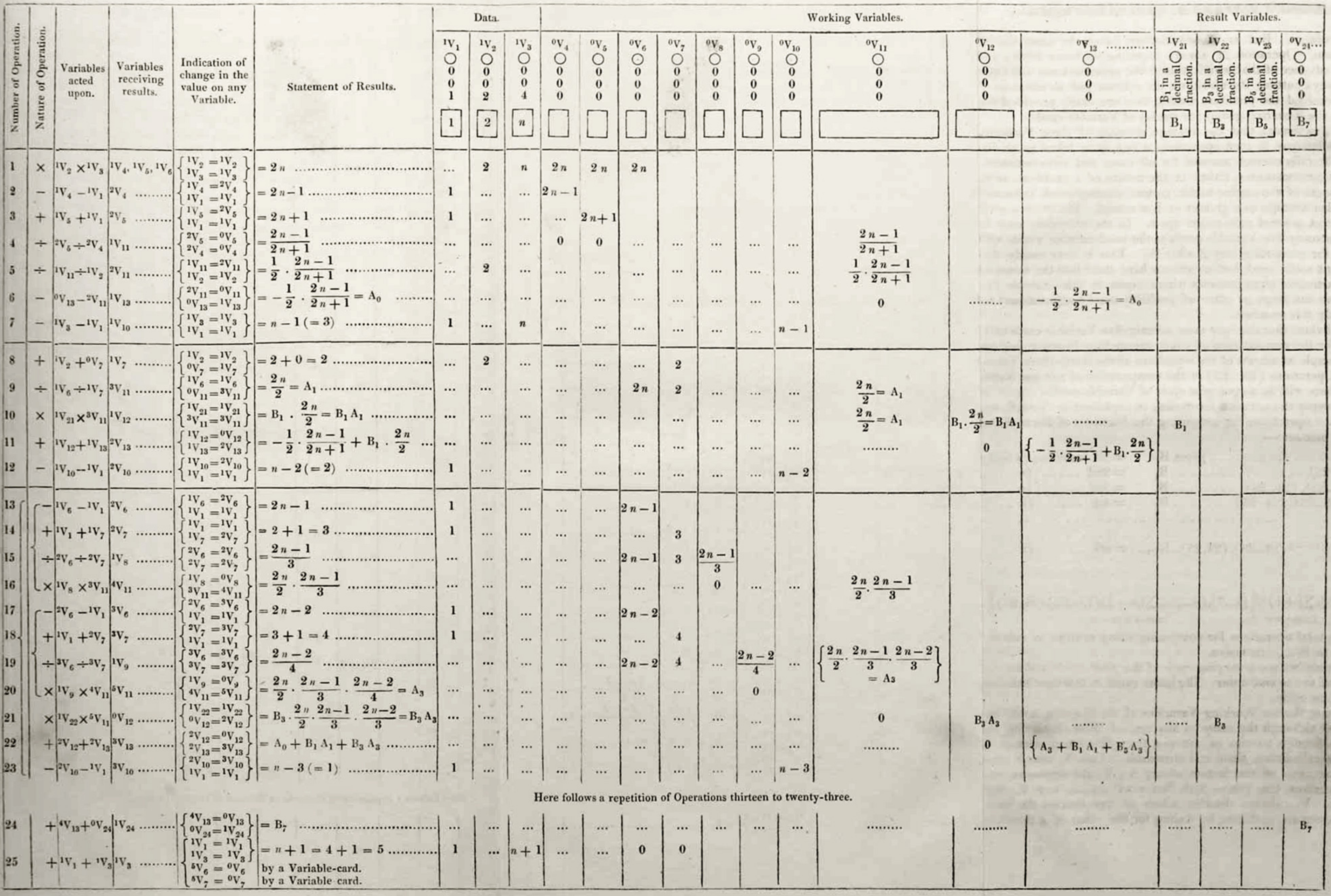


Unit	Bytes	Which is approximately
1 kilobyte (KB)	1024 bytes	10^{3} (1024 bytes exactly)
1 megabyte (MB)	1024 kilobytes	10^{6} (1,000,000 bytes)
1 gigabyte (GB)	1024 megabytes	10^{9} (1,000,000,000 bytes)
1 terabyte (TB)	1024 gigabytes	10^{12} (1,000,000,000,000 bytes)
1 petabyte (PB)	1024 terabytes	10^{15} (1,000,000,000,000,000 bytes)
1 exabyte (EB)	1024 petabytes	10^{18} (1,000,000,000,000,000,000 bytes)
1 zettabyte (ZB)	1024 exabytes	10^{21} (1,000,000,000,000,000,000,000 bytes)

Perkembangan Bahasa Pemograman

Tahun	Siapa	Apa
1800	Joseph Marie Charles	Membuat punch card untuk mengontrol mesin tenun
1842	Ada Lovelace	Membuat analisa untuk menyelesaikan persamaan Bernoulli
1936	Alan Turing	Membuat mesin Turing
	Alonzo Church	Berhasil membuat program lambda calculus dengan menggunakan mesin Turing
1957	John Backus	Memimpin pengembangan bahasa FORTRAN. Bahasa yang digunakan untuk High-Performance Computing. Masih digunakan sampai sekarang.

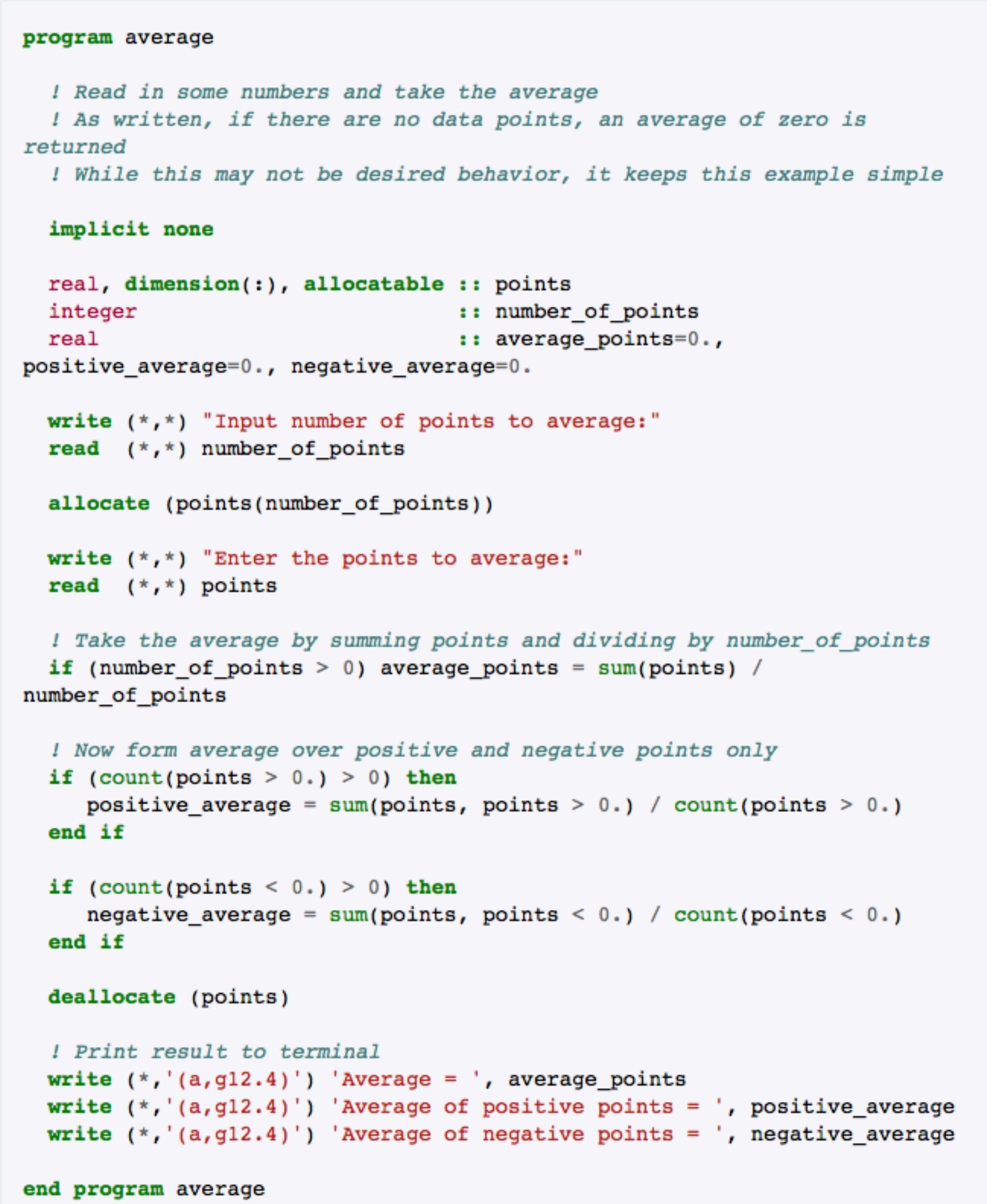
Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)



Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.										Result Variables.			
						$^1V_1$ ○ 0 0 0 1	$^1V_2$ ○ 0 0 0 2	$^1V_3$ ○ 0 0 0 4	$^0V_4$ ○ 0 0 0 0	$^0V_5$ ○ 0 0 0 0	$^0V_6$ ○ 0 0 0 0	$^0V_7$ ○ 0 0 0 0	$^0V_8$ ○ 0 0 0 0	$^0V_9$ ○ 0 0 0 0	$^0V_{10}$ ○ 0 0 0 0	$^0V_{11}$ ○ 0 0 0 0	$^0V_{12}$ ○ 0 0 0 0	$^0V_{13}$ .... ○ 0 0 0 0	$^1V_{21}$ ○ B_1 in a decimal fraction.	$^1V_{22}$ ○ B_3 in a decimal fraction.	$^1V_{23}$ ○ B_5 in a decimal fraction.	$^0V_{24}$... ○ 0 0 0 0
						$\boxed{1}$	$\boxed{2}$	$\boxed{n}$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{\ }$	$\boxed{B_1}$	$\boxed{B_3}$	$\boxed{B_5}$	$\boxed{B_7}$
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{\begin{matrix} ^1V_2 = {}^1V_2 \\ ^1V_3 = {}^1V_3 \end{matrix}\right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{\begin{matrix} ^1V_4 = {}^2V_4 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{\begin{matrix} ^1V_5 = {}^2V_5 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n + 1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{\begin{matrix} ^2V_5 = {}^0V_5 \\ ^2V_4 = {}^0V_4 \end{matrix}\right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{\begin{matrix} ^1V_{11} = {}^2V_{11} \\ ^1V_2 = {}^1V_2 \end{matrix}\right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix} ^2V_{11} = {}^0V_{11} \\ ^0V_{13} = {}^1V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{\begin{matrix} ^1V_3 = {}^1V_3 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 1\ (= 3)$	1	...	n	...	...	...	...	...	...	$n - 1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{\begin{matrix} ^1V_2 = {}^1V_2 \\ ^0V_7 = {}^1V_7 \end{matrix}\right\}$	$= 2 + 0 = 2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{\begin{matrix} ^1V_6 = {}^1V_6 \\ ^0V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix} ^1V_{21} = {}^1V_{21} \\ ^3V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix} ^1V_{12} = {}^0V_{12} \\ ^1V_{13} = {}^2V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{\begin{matrix} ^1V_{10} = {}^2V_{10} \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 2\ (= 2)$	1	...	...	...	...	...	...	...	...	$n - 2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{\begin{matrix} ^1V_6 = {}^2V_6 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{\begin{matrix} ^1V_1 = {}^1V_1 \\ ^1V_7 = {}^2V_7 \end{matrix}\right\}$	$= 2 + 1 = 3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{\begin{matrix} ^2V_6 = {}^2V_6 \\ ^2V_7 = {}^2V_7 \end{matrix}\right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix} ^1V_8 = {}^0V_8 \\ ^3V_{11} = {}^4V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{\begin{matrix} ^2V_6 = {}^3V_6 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n - 2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{\begin{matrix} ^2V_7 = {}^3V_7 \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 3 + 1 = 4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{\begin{matrix} ^3V_6 = {}^3V_6 \\ ^3V_7 = {}^3V_7 \end{matrix}\right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\begin{matrix} \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} \\ = A_3 \end{matrix}\right\}$						
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix} ^1V_9 = {}^0V_9 \\ ^4V_{11} = {}^5V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix} ^1V_{22} = {}^1V_{22} \\ ^0V_{12} = {}^2V_{12} \end{matrix}\right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	....	......	B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix} ^2V_{12} = {}^0V_{12} \\ ^2V_{13} = {}^3V_{13} \end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{A_3 + B_1 A_1 + B_3 A_3\right\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{\begin{matrix} ^2V_{10} = {}^3V_{10} \\ ^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n - 3\ (= 1)$	1	...	...	...	...	...	...	...	...	$n - 3$							
					Here follows a repetition of Operations thirteen to twenty-three.																	
24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix} ^4V_{13} = {}^0V_{13} \\ ^0V_{24} = {}^1V_{24} \end{matrix}\right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	....	....	....	......	......	......	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\left\{\begin{matrix} ^1V_1 = {}^1V_1 \\ ^1V_3 = {}^1V_3 \\ ^5V_6 = {}^0V_6 \\ ^5V_7 = {}^0V_7 \end{matrix}\right\}$	$= n + 1 = 4 + 1 = 5$ by a Variable-card. by a Variable card.	1	...	$n + 1$	...	...	0	0										

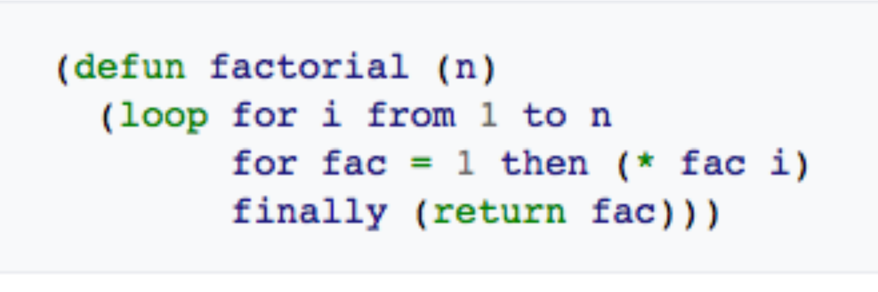
Fortran dari waktu ke waktu

1. Fortran
2. Fortran II
3. Fortran III
4. IBM 1401 Fortran
5. Fortran IV
6. Fortran 66
7. Fortran 77
8. Fortran 90
9. Fortran 95
10. Fortran 2003
11. Fortran 2008
12. Fortran 2018



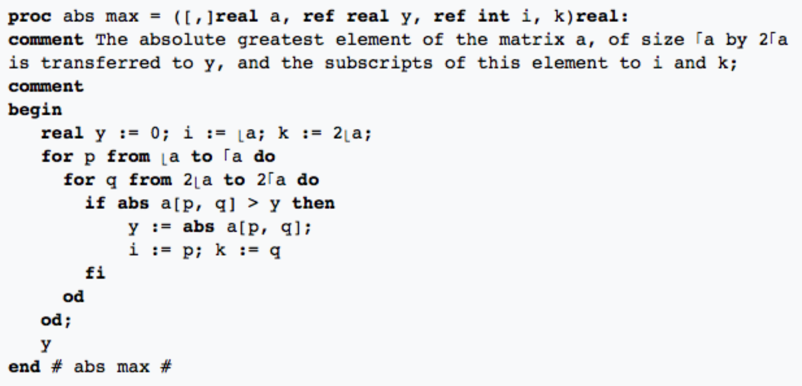
```
program average

  ! Read in some numbers and take the average
  ! As written, if there are no data points, an average of zero is
returned
  ! While this may not be desired behavior, it keeps this example simple

  implicit none

  real, dimension(:), allocatable :: points
  integer                         :: number_of_points
  real                            :: average_points=0.,
positive_average=0., negative_average=0.

  write (*,*) "Input number of points to average:"
  read  (*,*) number_of_points

  allocate (points(number_of_points))

  write (*,*) "Enter the points to average:"
  read  (*,*) points

  ! Take the average by summing points and dividing by number_of_points
  if (number_of_points > 0) average_points = sum(points) /
number_of_points

  ! Now form average over positive and negative points only
  if (count(points > 0.) > 0) then
     positive_average = sum(points, points > 0.) / count(points > 0.)
  end if

  if (count(points < 0.) > 0) then
     negative_average = sum(points, points < 0.) / count(points < 0.)
  end if

  deallocate (points)

  ! Print result to terminal
  write (*,'(a,g12.4)') 'Average = ', average_points
  write (*,'(a,g12.4)') 'Average of positive points = ', positive_average
  write (*,'(a,g12.4)') 'Average of negative points = ', negative_average

end program average
```

Perkembangan Bahasa Pemograman



```
(defun factorial (n)
  (loop for i from 1 to n
        for fac = 1 then (* fac i)
        finally (return fac)))
```

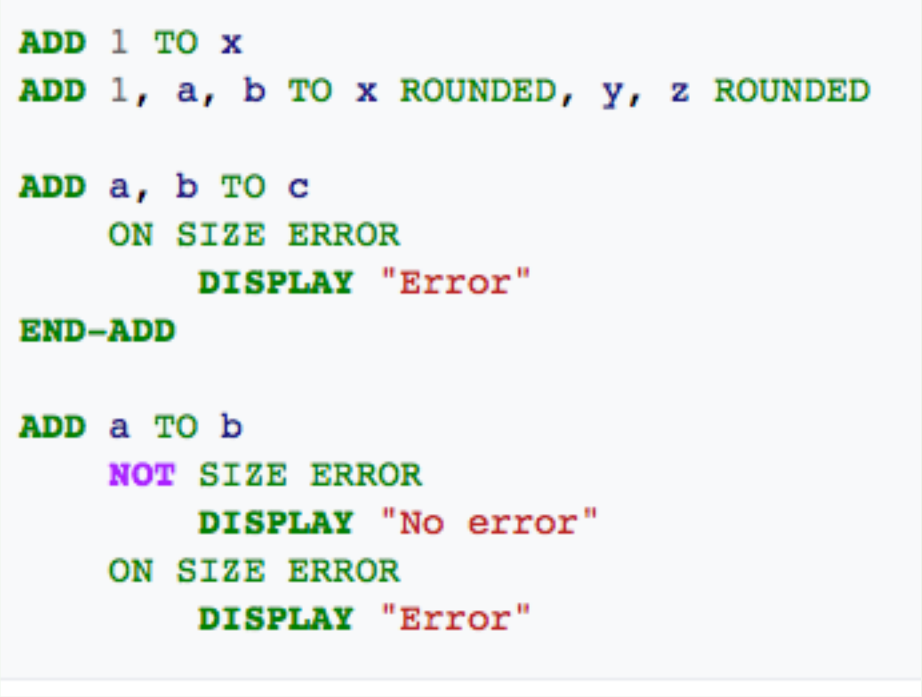
Tahun	Siapa	Apa
1958	John McCarthy	LISt Processor (LISP), dipengaruhi oleh lambda calculus dari Alonzo Church. Common LISP merupakan yang paling sering digunakan.
1958	Kumpulan ilmuwan (ETH Zurich)	Membuat bahasa pemograman Algorithmic Language (ALGOL)
1959	Grace Hopper	Membuat bahasa pemograman yang lebih terbaca (COBOL : Commong Business Oriented Language).



```
proc abs max = ([,]real a, ref real y, ref int i, k)real:
comment The absolute greatest element of the matrix a, of size ⌈a by 2⌈a
is transferred to y, and the subscripts of this element to i and k;
comment
begin
   real y := 0; i := ⌊a; k := 2⌊a;
   for p from ⌊a to ⌈a do
     for q from 2⌊a to 2⌈a do
       if abs a[p, q] > y then
           y := abs a[p, q];
           i := p; k := q
       fi
     od
   od;
   y
end # abs max #
```

Cobol dari waktu ke waktu

1. Cobol-60
2. Cobol-61/65
3. Cobol-68
4. Cobol-74
5. Cobol-85
6. Cobol 2002
7. Cobol 2014
8. Terancam punah

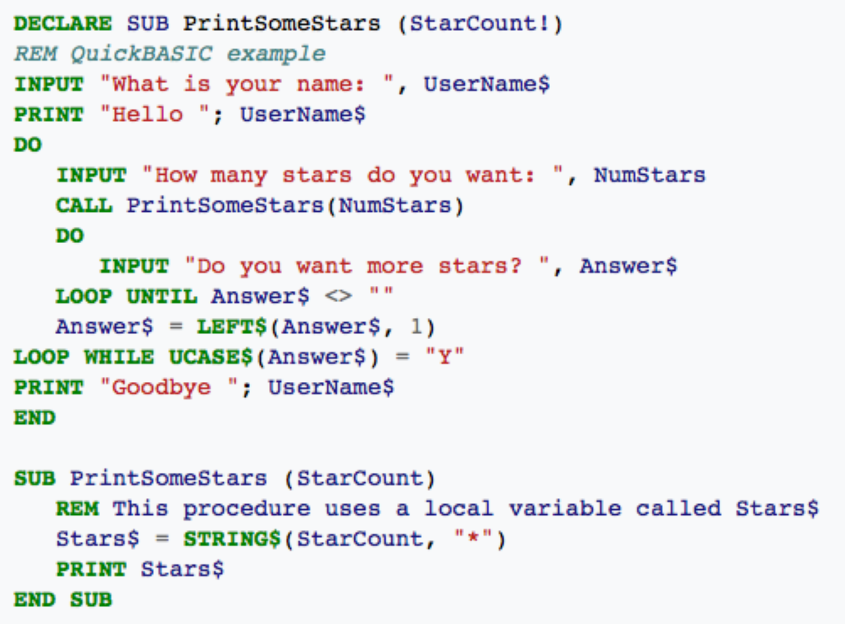


```
ADD 1 TO x
ADD 1, a, b TO x ROUNDED, y, z ROUNDED

ADD a, b TO c
    ON SIZE ERROR
        DISPLAY "Error"
END-ADD

ADD a TO b
    NOT SIZE ERROR
        DISPLAY "No error"
    ON SIZE ERROR
        DISPLAY "Error"
```

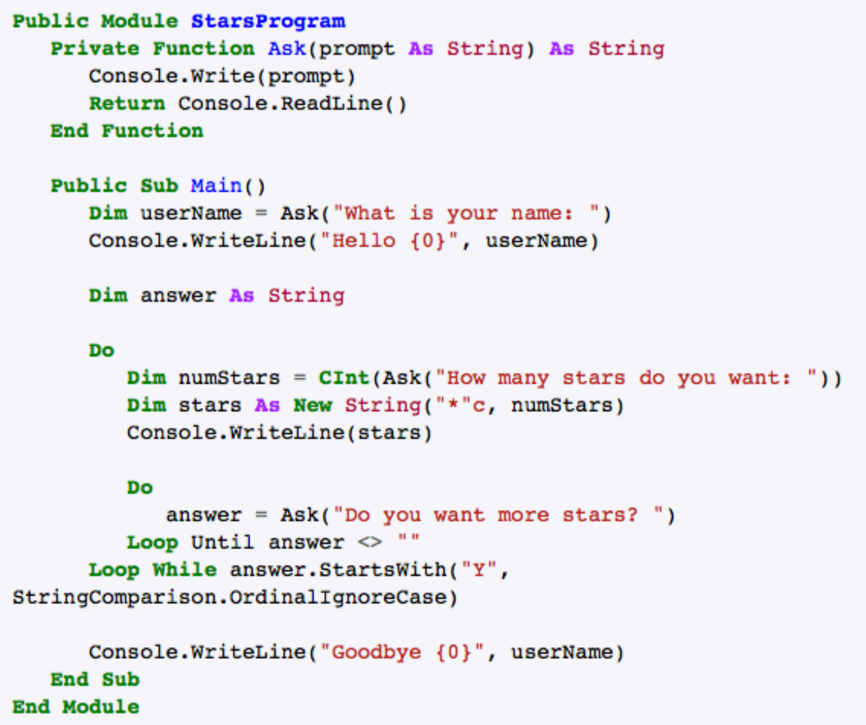
Perkembangan Bahasa Pemograman

Tahun	Siapa
1964	John G, Kemeny dan Thomas E. Kurtz

Beginner's All-purpose Symbolic Instruction Code (BASIC). Microsoft Visual Basic



```
DECLARE SUB PrintSomeStars (StarCount!)
REM QuickBASIC example
INPUT "What is your name: ", UserName$
PRINT "Hello "; UserName$
DO
    INPUT "How many stars do you want: ", NumStars
    CALL PrintSomeStars(NumStars)
    DO
        INPUT "Do you want more stars? ", Answer$
    LOOP UNTIL Answer$ <> ""
    Answer$ = LEFT$(Answer$, 1)
LOOP WHILE UCASE$(Answer$) = "Y"
PRINT "Goodbye "; UserName$
END

SUB PrintSomeStars (StarCount)
    REM This procedure uses a local variable called Stars$
    Stars$ = STRING$(StarCount, "*")
    PRINT Stars$
END SUB
```

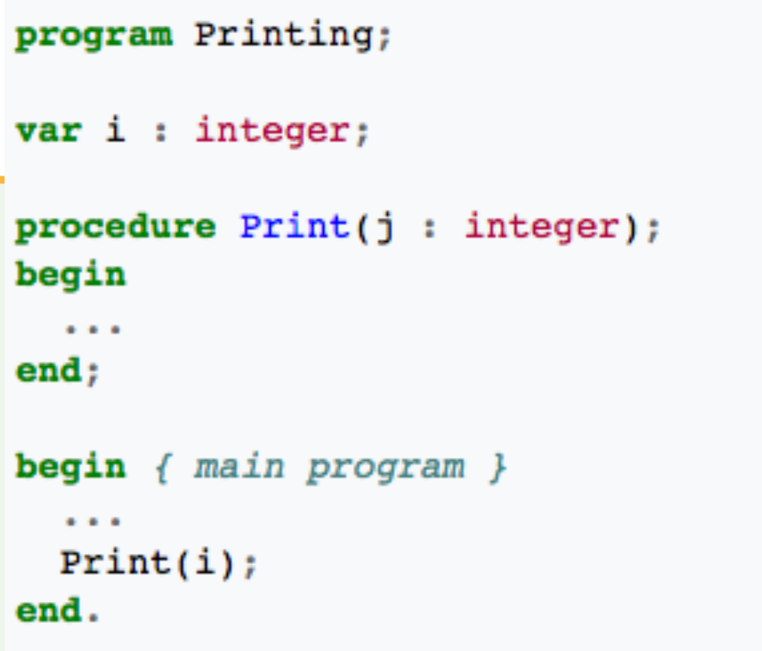


```
Public Module StarsProgram
    Private Function Ask(prompt As String) As String
        Console.Write(prompt)
        Return Console.ReadLine()
    End Function

    Public Sub Main()
        Dim userName = Ask("What is your name: ")
        Console.WriteLine("Hello {0}", userName)

        Dim answer As String

        Do
            Dim numStars = CInt(Ask("How many stars do you want: "))
            Dim stars As New String("*"c, numStars)
            Console.WriteLine(stars)

            Do
                answer = Ask("Do you want more stars? ")
            Loop Until answer <> ""
        Loop While answer.StartsWith("Y",
StringComparison.OrdinalIgnoreCase)

        Console.WriteLine("Goodbye {0}", userName)
    End Sub
End Module
```

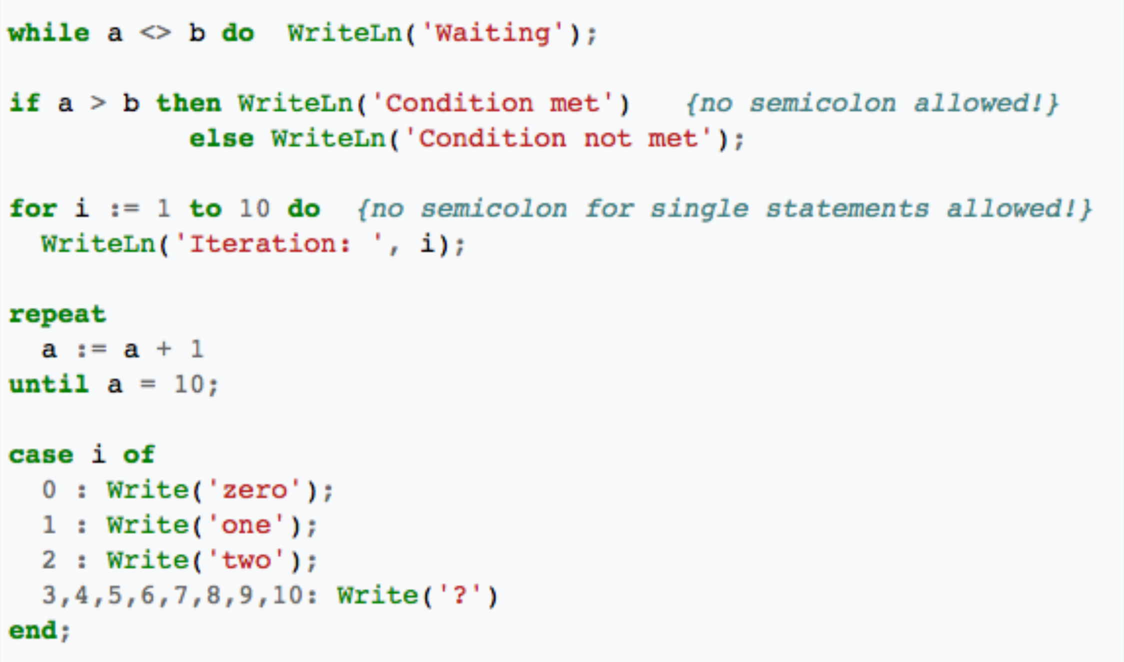
Perkembangan Bahasa Pemograman

Tahun	Siapa
1970	Niklaus Wirth

Pascal -> Turbo Pascal -> Object Pascal (basis Delphi)



```
program Printing;

var i : integer;

procedure Print(j : integer);
begin
  ...
end;

begin { main program }
  ...
  Print(i);
end.
```

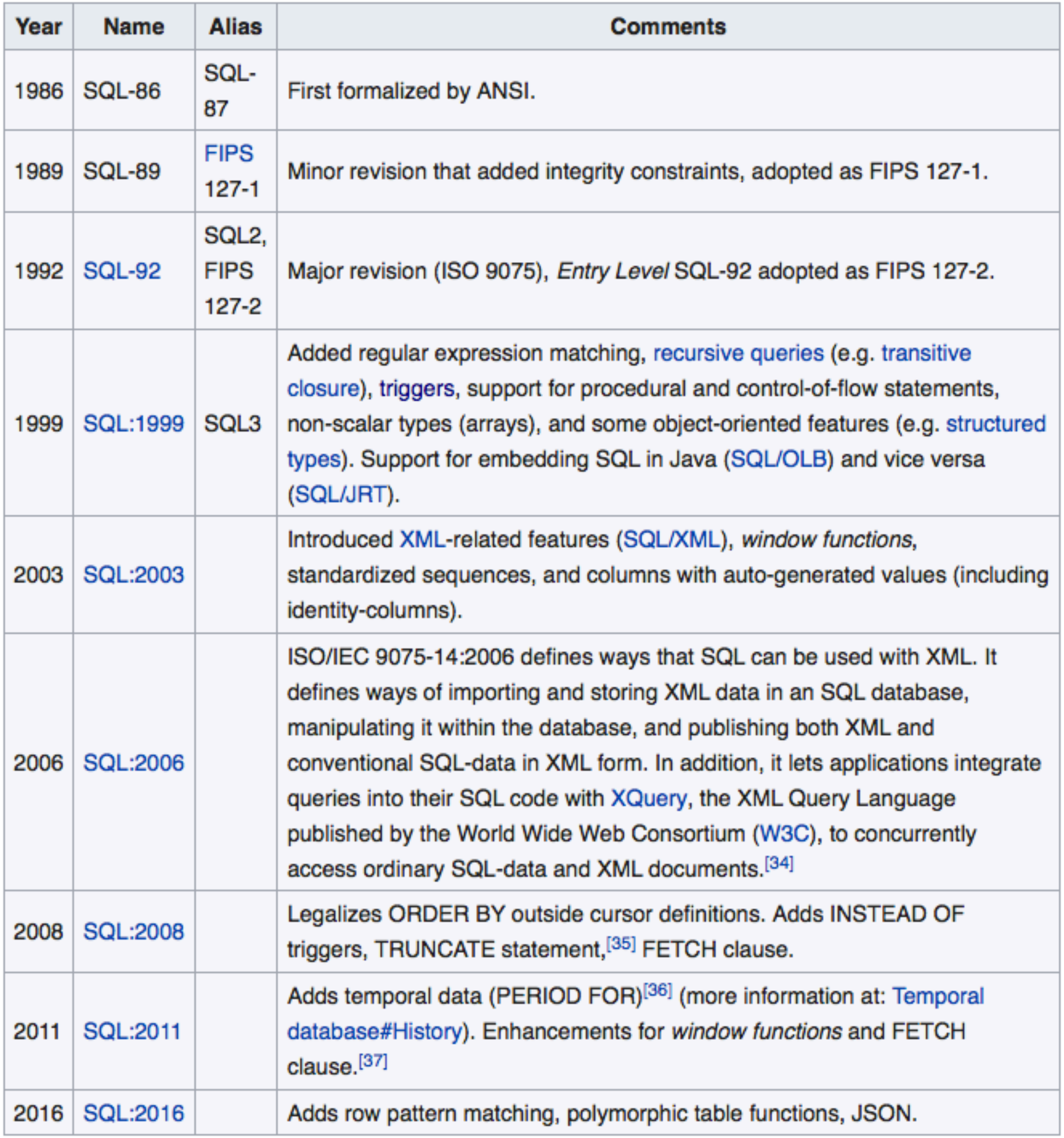


```
while a <> b do  WriteLn('Waiting');

if a > b then WriteLn('Condition met')   {no semicolon allowed!}
           else WriteLn('Condition not met');

for i := 1 to 10 do  {no semicolon for single statements allowed!}
  WriteLn('Iteration: ', i);

repeat
  a := a + 1
until a = 10;

case i of
  0 : Write('zero');
  1 : Write('one');
  2 : Write('two');
  3,4,5,6,7,8,9,10: Write('?')
end;
```

Perkembangan Bahasa Pemograman

Tahun	Siapa	Apa
1972	Dennis Ritchie	C -> ANSI C dan ISO C -> C99 -> C11 -> C18
1972	Alan Kay, dan Ingalls, Adele Goldberg	Smalltalk : object-oriented
1974	Donal D. Chamberlin, Raymond F. Boyce	Structured Query Language, digunakan untuk manipulasi data. Menjadi bahasa standar untuk basis data

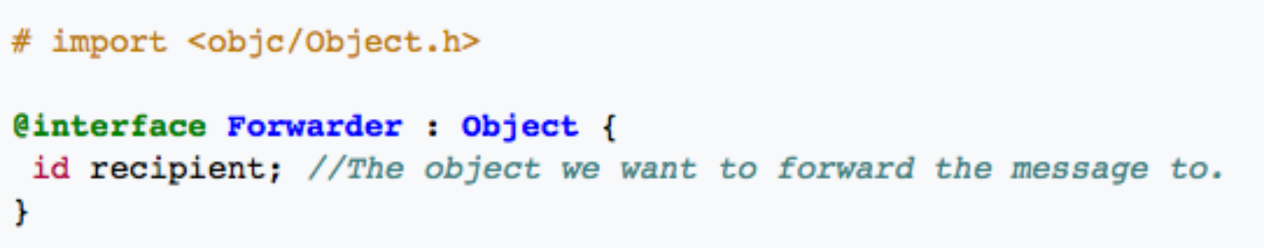
SQL dari waktu ke waktu



Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision that added integrity constraints, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types (arrays), and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice versa (SQL/JRT).
2003	SQL:2003		Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways that SQL can be used with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database, and publishing both XML and conventional SQL-data in XML form. In addition, it lets applications integrate queries into their SQL code with XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[34]
2008	SQL:2008		Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers, TRUNCATE statement,[35] FETCH clause.
2011	SQL:2011		Adds temporal data (PERIOD FOR)[36] (more information at: Temporal database#History). Enhancements for *window functions* and FETCH clause.[37]
2016	SQL:2016		Adds row pattern matching, polymorphic table functions, JSON.

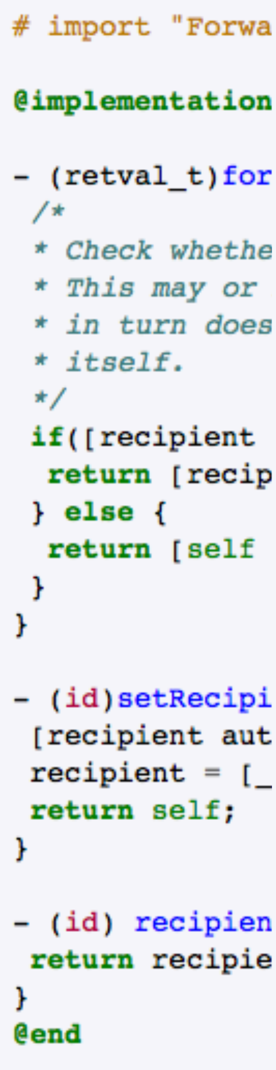
Perkembangan Bahasa Pemograman

Tahun	Siapa	Apa
1979	Bjarne Stroustrup	C++
1980	Jean Ichbiah	Ada. Target embedded dan real-time system
1984	Tom Love, Brad Cox	Objective C

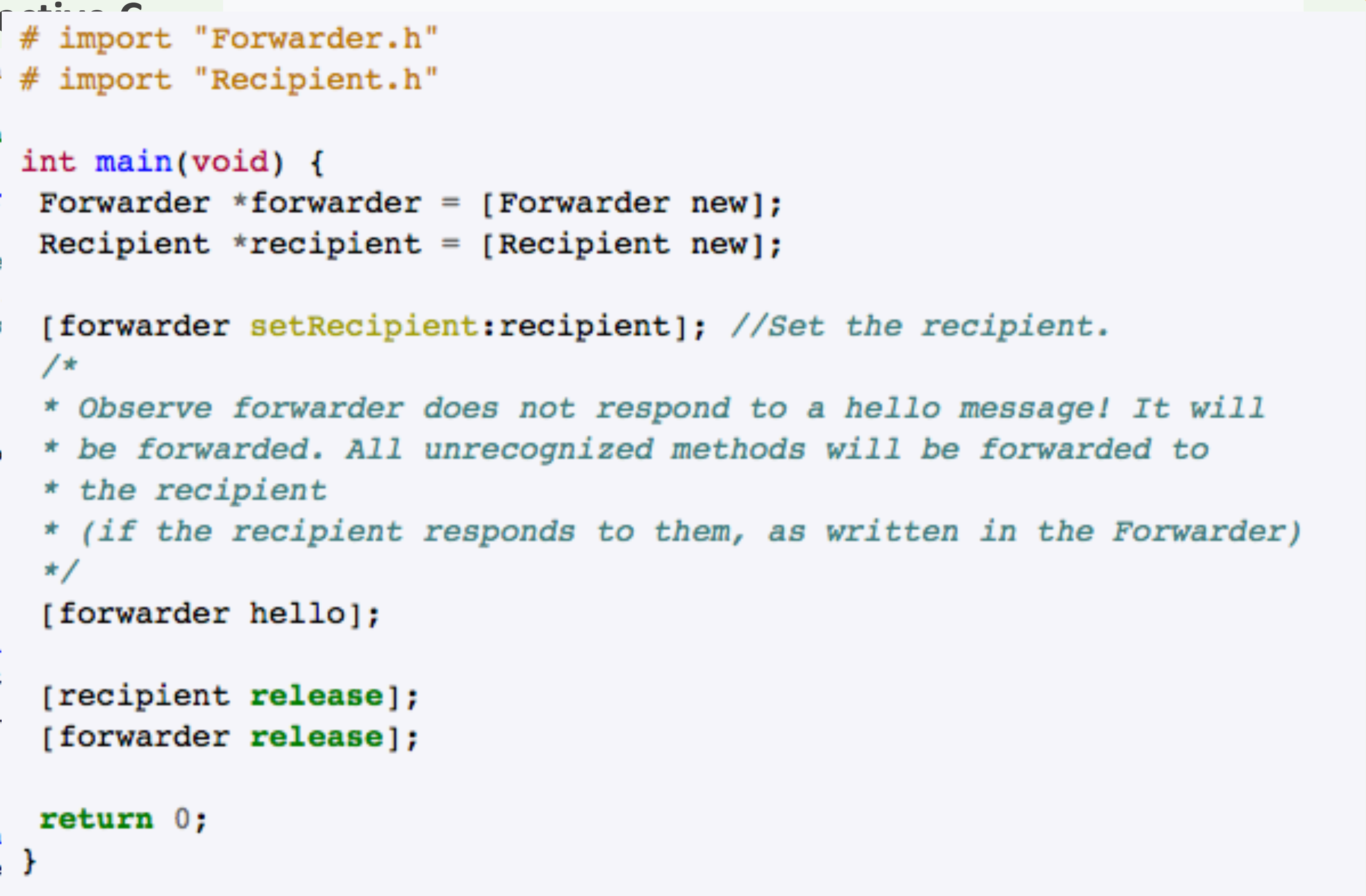


```
# import <objc/Object.h>

@interface Forwarder : Object {
 id recipient; //The object we want to forward the message to.
}
```

Obj



```
# import "Forwa

@implementation

- (retval_t)for
 /*
 * Check whethe
 * This may or
 * in turn does
 * itself.
 */
 if([recipient
  return [recip
 } else {
  return [self
 }
}

- (id)setRecipi
 [recipient aut
 recipient = [_
 return self;
}

- (id) recipien
 return recipie
}
@end
```



```
# import "Forwarder.h"
# import "Recipient.h"

int main(void) {
 Forwarder *forwarder = [Forwarder new];
 Recipient *recipient = [Recipient new];

 [forwarder setRecipient:recipient]; //Set the recipient.
 /*
 * Observe forwarder does not respond to a hello message! It will
 * be forwarded. All unrecognized methods will be forwarded to
 * the recipient
 * (if the recipient responds to them, as written in the Forwarder)
 */
 [forwarder hello];

 [recipient release];
 [forwarder release];

 return 0;
}
```

Perkembangan Bahasa Pemograman

Tahun	Siapa	Apa
1991	Guido van Rossum	Phyon
1995	James Gosling	Java
	Yukihiro Matsumotor	Ruby
	Brendan Eich	Javascript
	Rasmus Lerdorf	PHP

Java

Paradigma baru : Write once, run anywhere
Konsep Java Virtual Machine, bytecode

5 principles

1. It must be "simple, object-oriented, and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".
5. It must be "interpreted, threaded, and dynamic".

Ruby

I was talking with my colleague about the possibility of an object-oriented scripting language. I knew Perl (Perl4, not Perl5), but I didn't like it really, because it had the smell of a toy language (it still has). The object-oriented language seemed very promising. I knew Python then. But I didn't like it, because I didn't think it was a true object-oriented language — OO features appeared to be add-on to the language. As a language maniac and OO fan for 15 years, I really wanted a genuine object-oriented, easy-to-use scripting language. I looked for but couldn't find one. So I decided to make it.

Javascript

Bersama dengan HTML, CSS, menguasai web.
Bermula sebagai client-side berkembangan menjadi server-side (Node.js)

PHP (Personal Home Page/PHP: Hypertext Preporcessor)

Server-side scripting

Early PHP was not intended to be a new programming language, and grew organically, with Lerdorf noting in retrospect: "I don't know how to stop it, there was never any intent to write a programming language [...] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way." A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997

Perkembangan Bahasa Pemograman

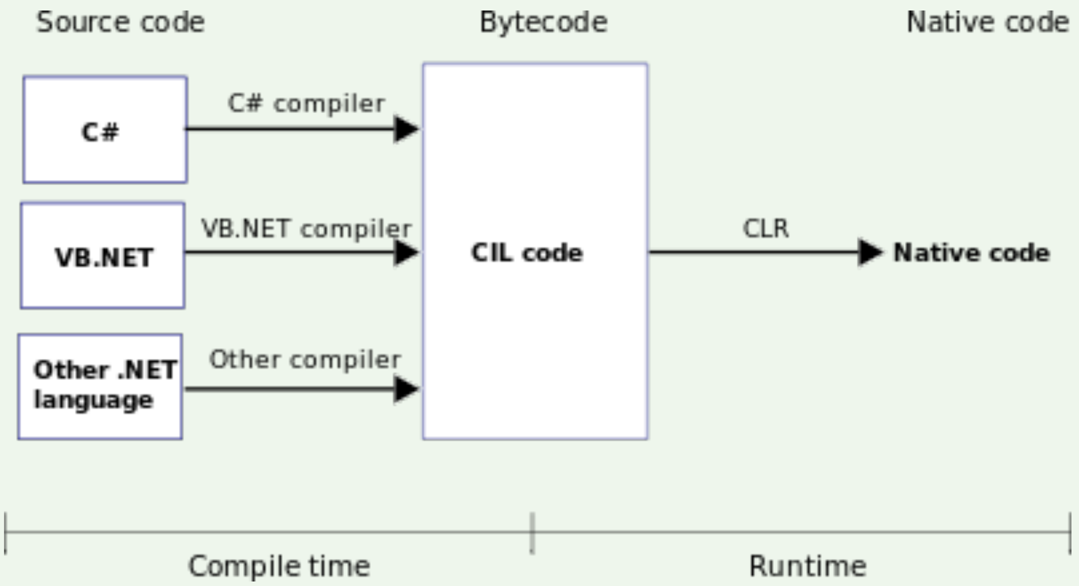
Tahun	Siapa	Apa
2000	Microsoft	C#
2004	James Strachan	Groovy
	Martin Odersky	Scala
2009	Robert Griesemer, Rob Pike, Ken Thompson	Go
2011	JetBrains	Kotlin
2014	Chris Lattner, Apple	Swift

C#

- The language is intended to be a simple, modern, general-purpose, object-oriented programming language.
- The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important.
- The language is intended for use in developing software components suitable for deployment in distributed environments.
- Portability is very important for source code and programmers, especially those already familiar with C and C++.
- Support for internationalization is very important.
- C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions.
- Although C# applications are intended to be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

C#

Common Language Runtime (CLR) : .NET virtual machine
Common Language Infrastructure (CLI)



Groovy, Scala, Kotlin

Groovy : bahasa scripting dan programming
Scala : procedural programming
Kotlin : digunakan untuk Android

dibuat menjadi bytecode, jalan di JVM

Swift

Pengganti Objective-C untuk pengembangan perangkat dari Apple

Go

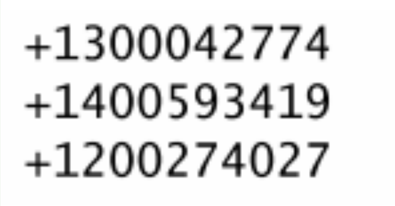
Bahasa pemograman yang mudah dari Google

Paradigma Programming

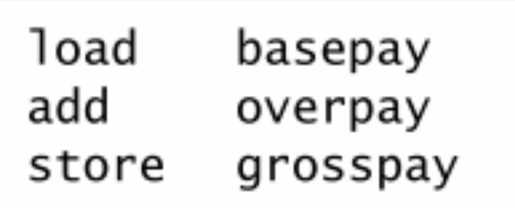
1. Imperative : programmer memberikan instruksi ke komputer bagaimana merubah kondisi. Terbagi dari :
   a. Procedural : grup prosedur/fungsi (C)
   b. Object-oriented : mengrup prosedur dalam kelas/objek
2. Declarative : programmer mendeklarasikan hasil akhir bukan cara merubah kondisi. Terdiri dari :
   a. Functional (Lisp)
   b. Logic (Prolog)
   c. Mathematical (Analytical)

Tipe Bahasa Pemograman

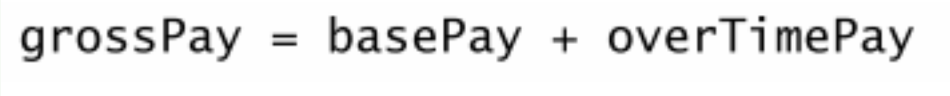
1. Bahasa mesin
2. Bahasa assembly
3. Bahasa tingkat tinggi



```
+1300042774
+1400593419
+1200274027
```



```
load     basepay
add      overpay
store    grosspay
```



```
grossPay = basePay + overTimePay
```

Pemrosesan Bahasa Tingkat Tinggi

1. Compiler : diubah dulu menjadi bahasa mesin dan dieksekusi
2. Intepreter : langsung dieksekusi

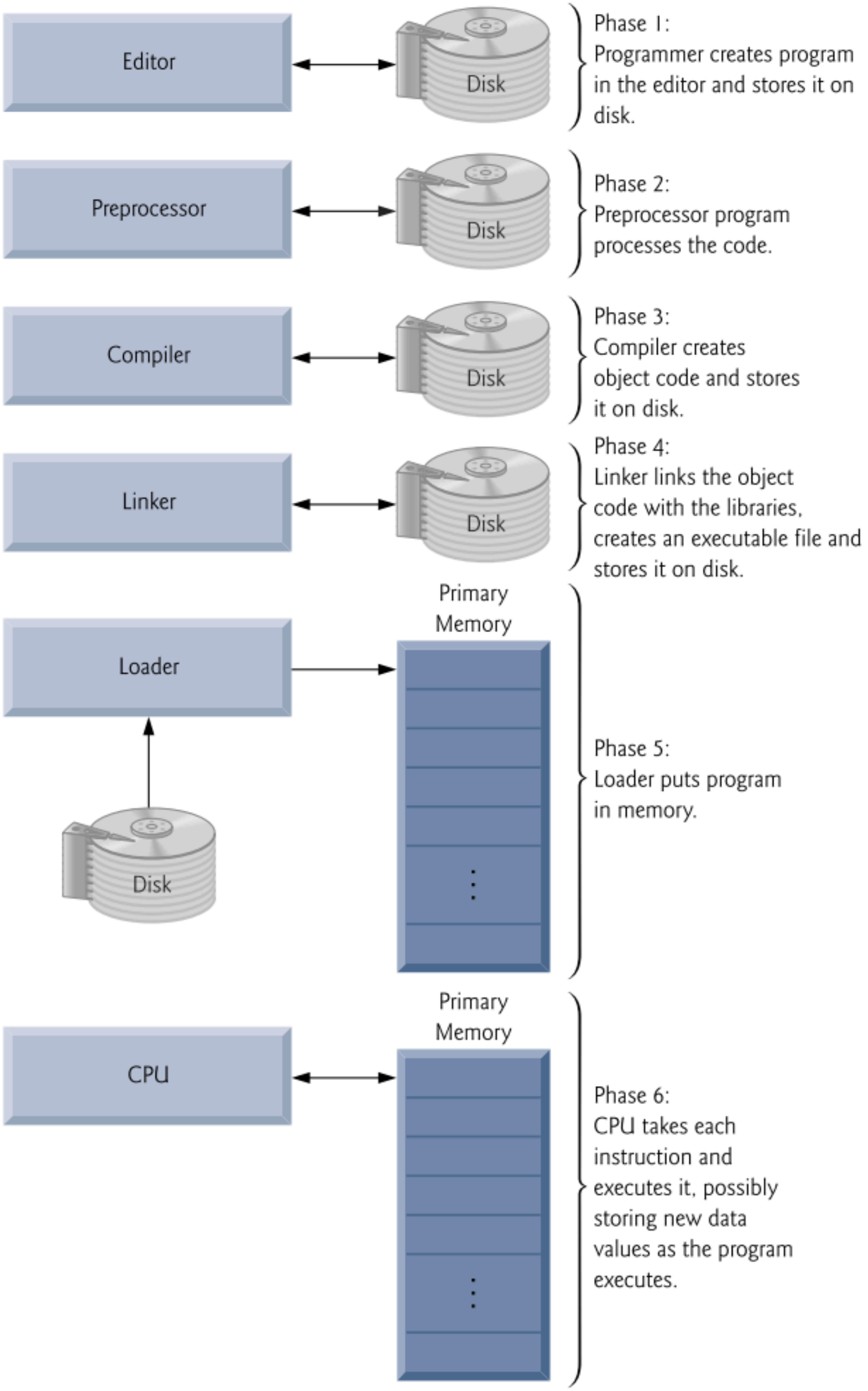
Bahasa Pemograman C

- Dibuat untuk performa tinggi

1. Sistem operasi : hampir semua sistem operasi dibuat menggunakan bahasa C
2. Embedded system (sistem tanam) : memastikan sistem berjalan dengan sangat cepat
3. Real time system : “mission-critical” application. Ketepatan waktu adalah kebutuhan
4. Communication system : delay bukan pilihan

- Standarisasi

Bahasa Pemograman C



Phase 1:
Programmer creates program in the editor and stores it on disk.

Phase 2:
Preprocessor program processes the code.

Phase 3:
Compiler creates object code and stores it on disk.

Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk.

Phase 5:
Loader puts program in memory.

Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Ayat Inspirasi

Allah mensyariatkan bagimu tentang (pembagian pusaka untuk) anak-anakmu. Yaitu: bahagian seorang anak lelaki sama dengan bahagian dua orang anak perempuan; dan jika anak itu semuanya perempuan lebih dari dua, maka bagi mereka dua pertiga dari harta yang ditinggalkan; jika anak perempuan itu seorang saja, maka ia memperoleh separo harta. Dan untuk dua orang ibu-bapak, bagi masing-masingnya seperenam dari harta yang ditinggalkan, jika yang meninggal itu mempunyai anak; jika orang yang meninggal tidak mempunyai anak dan ia diwarisi oleh ibu-bapaknya (saja), maka ibunya mendapat sepertiga; jika yang meninggal itu mempunyai beberapa saudara, maka ibunya mendapat seperenam. (Pembagian-pembagian tersebut di atas) sesudah dipenuhi wasiat yang ia buat atau (dan) sesudah dibayar utangnya. (Tentang) orang tuamu dan anak-anakmu, kamu tidak mengetahui siapa di antara mereka yang lebih dekat (banyak) manfaatnya bagimu. Ini adalah ketetapan dari Allah. Sesungguhnya Allah Maha Mengetahui lagi Maha Bijaksana. QS An-Nisaa : 11

Perhitungan warisan merupakan hal yang krusial dalam Islam. Penggunaan aplikasi dapat meminimalisir kesalahan perhitungan. Ini menjadikan inspirasi untuk MK Dasar Pemrograman

Referensi

C How To Program with an introduction to C++ (2016), P. Deitel, H. Deitel

DOA SESUDAH BELAJAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

اَللَّهُمَّ أَرِنَا الْحَقَّ حَقًّا وَارْزُقْنَا اتِّبَاعَهُ وَأَرِنَا الْبَاطِلَ بَاطِلاً وَارْزُقْنَا اجْتِنَابَهُ

Ya Allah, Tunjukkanlah kepada kami kebenaran sehinggga kami dapat mengikutinya Dan tunjukkanlah kepada kami kejelekan sehingga kami dapat menjauhinya

Universitas ‘Aisyiyah
Yogyakarta